Surface Reconstruction from Multiple Views using Rational B-Splines

Matheen Siddiqui and Stan Sclaroff*
Image and Video Computing Group – Computer Science Dept.
Boston University – Boston, MA 02215

Abstract

A method for reconstructing 3D rational B-spline surfaces from multiple views is proposed. The method takes advantage of the projective invariance properties of rational B-splines. Given feature correspondences in multiple views, the 3D surface is reconstructed via a four step framework. First, corresponding features in each view are given an initial surface parameter value (s, t), and a 2D B-spline is fitted in each view. After this initialization, an iterative minimization procedure alternates between updating the 2D B-spline control points and re-estimating each feature's (s, t). Next, a non-linear minimization method is used to upgrade the 2D B-splines to 2D rational B-splines, and obtain a better fit. Finally, a factorization method is used to reconstruct the 3D B-spline surface given 2D B-splines in each view. This surface recovery method can be applied in both the perspective and orthographic case. The orthographic case allows the use of additional constraints in the recovery. Experiments with real and synthetic imagery demonstrate the efficacy of the approach for the orthographic case.

1 Introduction

Recovering models from multiple views is an area of great interest in computer vision. Algorithms for recovering 3D structure and camera motion have many practical applications, such as computer aided design, virtual reality, movie special effects, video coding, etc. Many previous structure from motion algorithms are general in the sense that they assume no prior knowledge about the scene. However, in practice, the scene typically contains structures with strong geometric regularities that can be used to constrain the estimation problem. Consequently, algorithms have been proposed for the special cases of planes [1, 11, 22, 26, 27, 31], piecewise planar models [3, 19], polygonal meshes [14], and quadric surfaces [8, 18].

In this paper we propose a method for reconstructing 3D rational B-spline surfaces from multiple views. As will be shown, we can exploit the the projective invariance properties of rational B-splines to gain a solution to the 3D surface estimation problem. The approach is demonstrated in a formulation for recovering a quadratic B-spline surface from point correspondences given in multiple views. The for-

mulation can be applied in both the perspective and orthographic case. The orthographic case allows the use of additional constraints in the recovery. Experiments with real and synthetic imagery demonstrate the efficacy of the approach for the orthographic case.

2 Related Work

Structure recovery algorithms can be grouped according to the complexity of their shape prior. Algorithms using no shape priors, recover 3D point locations directly. One example of this is due to [28], where image measurements are assembled into a large matrix which can then be factorized to reveal the underlying geometry. Others have extend this approach to the perspective case [21, 29]. In contrast with these batch processing approaches, others have developed methods that employ a recursive estimation theory (e.g., a Kalman filter) to estimate 3D point structure and camera parameters in an on-line fashion [2, 4, 6, 16, 20]. This approach allows for observation and process noise models to be directly incorporated into the reconstruction method.

Higher-level constraints of the structure have been incorporated into reconstruction methods as well. In particular, the special case of planar structure has been extensively researched, and there are methods that incorporate the resulting constraints [26]. In particular [31] has developed a closed-form solution to planar reconstruction from two views and [1] had extended the recursive framework of [2], to incorporate planar information. In [19] Sinclair focuses on grouping planar regions, while [3] focuses on direct reconstruction. Some methods segment the scene and determine planes for each patch [11, 22, 27].

More general surface representations in the form of meshes were explored by [14]. In these approaches the surface is found by minimizing an objective function relating the estimated surface with projections into estimated cameras. Surfaces have also been modeled as oriented particles or tiny planes with associated texture in [13, 15, 25]. While very general, mesh and particle representations tend to over parameterize smoothly curved surfaces.

In [8] and [18], methods have been proposed that can be used when objects are well-approximated by quadratic surfaces. In [8], the quadratic surface is estimated by relating the silhouette of the quadratic surfaces to the projected image. In a different approach, [18] examines the induced flow field of quadratic objects.

In this work we further develop the representation of

Report Documentation Page

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **2001**	2. REPORT TYPE	3. DATES COVERED **-**

4. TITLE AND SUBTITLE **Surface Reconstruction from Multiple Views using Rational B-Splines**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Office of Naval Research,One Liberty Center,875 North Randolph Street Suite 1425,Arlington,VA,22203-1995**	8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**8**	

smooth surfaces by using rational B-splines to represent 3D surfaces and their projections. Spline curve representations have been used extensively in computer vision in the context of contour tracking and contour pose recovery (e.g., [7, 5]). Splines have also been used in motion estimation [24] and image registration [23]. Our work is related to [23] where a depth field is computed from spline-based image registration. In that approach, the spline was used to find disparity maps from which dense depth maps were then computed. In this work, we instead focus on directly modeling surfaces in the scene as 3D rational B-splines.

3 Rational B-Splines

In this section we introduce relevant concepts in B-splines and define the notation used in this paper. For a detailed review of B-splines, readers are directed to [10].

A B-spline surface can be defined in terms of a $(m+1) \times (n+1)$ grid of *control points*, and a set of *blending functions* (interpolation functions). Given these, we define a point on the B-spline surface at a particular parameter value (s, t):

$$\mathbf{P}(s, t) = \sum_{i=0}^{m} \sum_{j=0}^{n} B_{i,d}(t) B_{j,d}(s) \mathbf{p}_{i,j} \tag{1}$$

where $\mathbf{p}_{i,j}$ is a control point, and $B_{i,d}$ and $B_{j,d}$ are blending functions in each parametric direction. The B-spline blending functions can be obtained via the Cox de Boor recursion formulas, i.e.:

$$B_{j,1}(s) = \begin{cases} 1, & \text{if } s_j \leq s < s_{j+1} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$\begin{aligned} B_{j,d}(s) = & \frac{s - s_j}{s_{j+d-1} - s_j} B_{j,d-1}(s) \\ & + \frac{s_{j+d} - s}{s_{j+d} - s_{j+1}} B_{j+1,d-1}(s). \end{aligned} \tag{3}$$

The polynomial order of the blending functions is defined by d. The influence of the blending functions is controlled by a knot vector, $[s_j]$ ($s_j \leq s_{j+1}$), which defines the region of influence of each blending function. In our implementation, we will set $d = 3$, $m = n = 4$, and employ a uniform knot vector: $[0, 0, 0, 1, 2, 3, 3, 3]$.

To make our problem formulation easier, we will need to deviate from the standard B-spline notation as follows. Given the set of $N = (m + 1) \times (n + 1)$ control points, we will rewrite (1):

$$\mathbf{P}(s, t) = \sum_{k=1}^{N} b_k \mathbf{P}_k, \tag{4}$$

where $\mathbf{P}_k = \mathbf{p}_{i,j}$ and $b_k = B_{i,d}(t) B_{j,d}(s)$, such that $i = 1 + \lfloor k/(n+1) \rfloor$ and $j = 1 + k \mod (n+1)$.

Using similar notation, rational B-splines take the form:

$$\mathbf{R}(s, t) = \frac{\sum_{k=1}^{N} b_k w_k \mathbf{P}_k}{\sum_{k=1}^{N} b_k w_k} \tag{5}$$

The weights, w_k are weight factors for the control points. The greater the value of w_k, the closer the surface is "pulled" towards the control point $\mathbf{P}_k$. One advantage of the rational B-spline representation is that it can be used to represent quadrics exactly. Another advantage is that they are invariant to perspective viewing transformations, as will be shown in Sec.3.1.

Rational B-splines can be written in terms of a homogeneous equation, with homogeneous control points:

$$\tilde{\mathbf{P}}_k = \begin{bmatrix} w_k \mathbf{P}_k \\ w_k \end{bmatrix} \tag{6}$$

In the homogeneous coordinate system, points on the surface are represented:

$$\tilde{\mathbf{R}}(s, t) = \sum_{k=1}^{N} b_k \tilde{\mathbf{P}}_k. \tag{7}$$

3.1 Projection of Rational B-Splines

An important benefit in using rational B-splines to represent surfaces becomes apparent when we consider the projection of a 3D surface point onto the image plane via a 3×4 projection matrix $\mathcal{P}$. A point on the rational B-spline $\tilde{\mathbf{R}}(s, t)$ is related to the projected image point $\mathbf{u}(s, t)$ via:

$$\begin{bmatrix} \mathbf{u}(s, t) \\ 1 \end{bmatrix} \simeq \mathcal{P} \tilde{\mathbf{R}}(s, t) \tag{8}$$

Here $\simeq$ is defined to be equality up to scale. From (8) it follows:

$$\begin{bmatrix} \mathbf{u}(s, t) \\ 1 \end{bmatrix} \simeq \mathcal{P} \sum_{k=1}^{N} b_k \tilde{\mathbf{P}}_k \tag{9}$$

$$\simeq \sum_{k=1}^{N} b_k \mathcal{P} \tilde{\mathbf{P}}_k \tag{10}$$

$$\simeq \sum_{k=1}^{N} b_k \tilde{\mathbf{P}}'_k \tag{11}$$

From (11) we see that to project a rational spline we simply project the 3D surface's (homogeneous) control points to obtain a set of projected (homogeneous) control points for the 2D surface (i.e., $\tilde{\mathbf{P}}'_k$). The projected surface points are generated using the same blending coefficients (b_k).

This relation suggests a method for reconstructing a B-spline surface given multiple views. In particular, to reconstruct the surface we simply need to recover the projected

control points, $\tilde{\mathbf{P}}'_k$, of the 2D rational spline in each view. Since these control points are in principal no different than other points we can employ traditional structure from motion methods (e.g., [21, 28, 30]) to obtain the corresponding 3D B-spline control points $\tilde{\mathbf{P}}_k$ and the camera parameters if desired.

4 Surface Recovery

Given feature correspondences in multiple views, the goal will be to reconstruct the 3D B-spline surface. It is assumed that the image location of M features and their correspondence in each view are given as input to our system. For a particular view, the location of the ith feature will be denoted by $\mathbf{u}_i = [u_i, v_i]^T$. To reduce the complexity of the surface reconstruction, the order of the spline surface and the knot vector are assumed given. As mentioned earlier, in our implementation we will set the order of the spline to be $d = 3$, and $m = n = 4$, and we will use a uniform knot vector. The number of control points $N = (m + 1) \times (n + 1) = 25$. We must therefore recover the N control point locations, as well as the surface point parameters (s_i, t_i) for each of M feature points.

As will be detailed in the rest of this section, the 3D surface will be reconstructed via a four step framework. First, corresponding features in each view are given an initial surface parameter value (s_i, t_i), and a 2D B-spline is fitted in each view. After this initialization, an iterative minimization procedure alternates between updating the 2D B-spline control points and re-estimating each feature's (s_i, t_i). Next, a nonlinear minimization method is used to upgrade the 2D B-splines to 2D rational B-splines, and obtain a better fit. Finally, a factorization method is used to reconstruct the 3D rational B-spline surface given 2D B-splines in each view.

4.1 Initializing (s_i, t_i)

Given a collection of features, an initial surface parameter (s_i, t_i) must be assigned for each feature. The ith feature is assigned the same (s_i, t_i) in all views. The estimate of each feature's surface parameter will be improved via an iterative procedure later. An initial (s_i, t_i) parameter value is assigned for each feature in the first view. Corresponding features in the remaining views inherit the parameter assigned in the first view.

The surface point parameters in the first view are found by placing a rectangular grid of regularly spaced 2D control points with $w_k = 1$ over the feature points in the first image. The overall size of the grid is determined by finding the minimum bounding box for the features. This results in a set of control points of the form $\tilde{\mathbf{P}}'_k = [\tilde{u}_k, \tilde{v}_k, 1]^T$, where $(\tilde{u}_k, \tilde{v}_k)$ is the image location for the kth control point.

The surface point parameters (s_i, t_i) can then be found by searching over the B-spline parameter space. In our implementation, this is done by sampling uniformly in $s - t$ space and choosing the sample $\mathbf{R}(s_l, t_l)$ that corresponds to the point closest to the feature point, $\mathbf{u}_i$:

$$s_i^*, t_i^* = \arg\min_{s_l, t_l} \|\mathbf{u}_i - \mathbf{R}(s_l, t_l)\| \qquad (12)$$

Sampling is then performed at a higher resolution around the chosen estimate, (s_i^*, t_i^*) to find an improved estimate that is closer to the feature via (12). This is repeated until the distance between the B-spline sample and the feature is within a specified threshold.

4.2 Refining Control Points of 2D B-Spline

Given an initial estimate of the features' parameter values (s_i, t_i), the control points in each of the views are estimated. In this step, the control points are recovered independently for each view, and the third component of the control points is fixed $w_k = 1$. For a given (s_i, t_i), the point on the B-spline can be computed:

$$\tilde{\mathbf{R}}(s_i, t_i)^T = \underbrace{[b_{i,1}, b_{i,2}, \ldots, b_{i,N}]}_{\mathbf{b}_i^T} \begin{bmatrix} \tilde{\mathbf{P}}'^T_1 \\ \tilde{\mathbf{P}}'^T_2 \\ \vdots \\ \tilde{\mathbf{P}}'^T_N \end{bmatrix}, \qquad (13)$$

where $\mathbf{b}_i$ is the vector of blending coefficients for (s_i, t_i).

Ideally we would like the feature point observations to match their assigned $\mathbf{R}(s_i, t_i)$. If this were the case the observations would be related to the control points via:

$$\begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \\ \vdots & \vdots \\ u_M & v_M \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1^T \\ \mathbf{b}_2^T \\ \vdots \\ \mathbf{b}_M^T \end{bmatrix} \begin{bmatrix} \tilde{u}_1 & \tilde{v}_1 \\ \tilde{u}_2 & \tilde{v}_2 \\ \vdots & \vdots \\ \tilde{u}_N & \tilde{v}_N \end{bmatrix}. \qquad (14)$$

This can be rewritten in the standard matrix vector form if we denote:

$$\mathbf{y} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_M \\ v_1 \\ v_2 \\ \vdots \\ v_M \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \vdots \\ \tilde{u}_N \\ \tilde{v}_1 \\ \tilde{v}_2 \\ \vdots \\ \tilde{v}_N \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{b}_1^T \\ \mathbf{b}_2^T \\ \vdots \\ \mathbf{b}_M^T \end{bmatrix}. \qquad (15)$$

Then (14) becomes

$$\mathbf{y} = \hat{\mathbf{B}}\mathbf{x} \qquad (16)$$

where:

$$\hat{\mathbf{B}} = \begin{bmatrix} \mathbf{B} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{bmatrix}. \qquad (17)$$

Since (16) will not be satisfied exactly due to measurement noise and error in assigning (s_i, t_i), we seek the solution that minimizes the mean squared error.

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} \|\mathbf{y} - \hat{\mathbf{B}}\mathbf{x}\|^2 \qquad (18)$$

which can be found using conjugate gradient descent [17].

4.3 Updating (s_i, t_i)

With updated control points we can proceed to improve each feature's assigned surface parameter, (s_i, t_i). Recall that corresponding points have the same parameters in all views. In refining our estimate of a feature's (s_i, t_i), we seek to minimize the sum of the distance between the hypothesized point and the observed feature location over all views:

$$s_i^*, t_i^* = \arg \min_{s_i, t_i} \sum_{j=1}^{N_v} \|\mathbf{u}_i^{(j)} - \mathbf{R}^{(j)}(s_i, t_i)\|^2 \qquad (19)$$

where N_v is the number of views, $\mathbf{u}_i^{(j)}$ is the observed feature location in the jth view, and $\mathbf{R}^{(j)}(s_i, t_i)$ is the spline predicted point location for that view. This can be solved efficiently using the previous estimate (s_i, t_i) as starting point in a minimization algorithm that is based on golden section search and parabolic interpolation [12].

Given improved estimates of (s_i, t_i) obtained through the nonlinear minimization, we can then refine the estimates of the control points in each view as described in Sec. 4.2. As suggested in [9], these two steps can be applied alternately, first updating the B-spline control points and then re-estimating each feature's (s_i, t_i). In our experience, we have observed that three or four iterations are needed for reasonable convergence.

4.4 Upgrade to Rational B-Splines

Recall that up until this point, the control point weights have all been set $w_k = 1$. Estimates of the the 2D spline control points have been obtained in each view, keeping the weights fixed as described above. We can now use a nonlinear minimization technique to further fit the spline in particular view, by allowing the weights to vary. For a particular view, each control point's weight is chosen to satisfy:

$$\tilde{P}_k^* = \arg \min_{\bar{P}_k} \sum_{i=1}^{M} \|\mathbf{u}_i - \mathbf{R}(s_i, t_i)\|^2 \qquad (20)$$

In practice, we can keep the point parameters while finding the weights. The solution is found using the Levenberg-Marquardt [17] nonlinear minimization procedure. The solution is computed separately for each view.

4.4.1 Special Case: Orthography

When the cameras are orthographic the project matrix takes the form:

$$\mathcal{P} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (21)$$

In this case, the w_k component of the control points of the 2D B-splines across all views are constrained to be the same. This additional constraint can be used when fitting the 2D rational B-splines. In this case, the objective function to be minimized should include the control points in all views simultaneously, since the kth control point in all views share the same w_k.

In addition, if the cameras are orthographic and it is known that the observed 3D surface can be well-approximated by a non-rational B-spline, then the "upgrade to rational B-splines" step can be omitted since the weights are known to all be $w_k = 1$.

4.5 Reconstruction

Given recovered 2D control points for rational B-splines in all views, we can employ standard factorization algorithms [28, 21] to recover the 3D shape and the camera matrices. The recovered 2D control points are assembled into a measurement matrix. This matrix is then factorized using SVD to reveal its *projective* structure. Following this, a homography is applied to both the cameras and the structure to transform the projective reconstruction to an Euclidean one.

4.5.1 Perspective Reconstruction

In the perspective case we assemble the homogenized control points into a measurement matrix that relates to the 3D B-spline surface control points and camera matrices as:

$$\begin{bmatrix} \tilde{\mathbf{P}}_1'^{(1)} & \cdots & \tilde{\mathbf{P}}_N'^{(1)} \\ \vdots & \ddots & \vdots \\ \tilde{\mathbf{P}}_1'^{(N_v)} & \cdots & \tilde{\mathbf{P}}_N'^{(N_v)} \end{bmatrix} = \begin{bmatrix} \mathcal{P}^{(1)} \\ \vdots \\ \mathcal{P}^{(N_v)} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{P}}_1 \ldots \tilde{\mathbf{P}}_N \end{bmatrix} \qquad (22)$$

Typically in perspective reconstructions of this form the homogenous component of each point is not known and must also be recovered. However, in our case the w_k's have already been found and can be incorporated directly. We can then assemble the recovered control points $\tilde{\mathbf{P}}_k'^{(j)}$ into the measurement matrix on the left-hand side of (22). The camera matrices and 3D control points are then recovered as described in [21].

4.5.2 Orthographic Reconstruction

Reconstruction in the orthographic case is similar to that of the perspective case. However, here we are interested in

the 2×3 representation of the camera matrices $\mathcal{M}$. To use this representation we need to substract off the centroid of the feature points in each view, which effectively sets the principal point to zero. Thus $\mathcal{M}$ is the upper left 2×3 submatrix of $\mathcal{P}$.

In this framework the reconstruction is found by relating the observations to the cameras and 3D control point locations through:

$$\left[\begin{array}{ccc} \hat{\mathbf{P}}'^{(1)}_1 & \cdots & \hat{\mathbf{P}}'^{(1)}_N \\ \vdots & \ddots & \vdots \\ \hat{\mathbf{P}}'^{(N_v)}_1 & \cdots & \hat{\mathbf{P}}'^{(N_v)}_N \end{array} \right] = \left[\begin{array}{c} \mathcal{M}^{(1)} \\ \vdots \\ \mathcal{M}^{(N_v)} \end{array} \right] [\mathbf{P}_1 \ldots \mathbf{P}_N], \tag{23}$$

where $\hat{\mathbf{P}}'^{(j)}_k$ are the recovered $[\tilde{u}^{(j)}_k, \tilde{v}^{(j)}_k]$ components of each control point $\tilde{\mathbf{P}}'^{(j)}_k$. The camera matrices and 3D control points are then recovered as described in [28]. The weights w_k of the control points are already known, as they were obtained in fitting the 2D rational splines (Sec. 4.4.1).

4.6 Minimum Number of Features and Views

In fitting the 2D rational B-Splines, there are a total of N control points with three degrees of freedom for each of N_v views. In addition, there are two surface parameters (s_i, t_i) for each of M observations. Each feature observation provides two constraints. Therefore, in general we need:

$$3 \times N \times N_v + 2 \times M \leq 2 \times M \times N_v. \tag{24}$$

In our implementation, biquadratic surfaces were used; therefore, in our case $N = 25$.

For the final reconstructed surface we have N 3D control points as well as the (s_i, t_i) for each of M observed points. Therefore, in the final reconstruction step we need:

$$4 \times N + 2 \times M \leq 2 \times M \times N_v. \tag{25}$$

For reconstruction we need the numbers of views and observed feature points to satisfy both (24) and (25).

5 Experiments

To evaluate our system, reconstructions were performed on real and synthetic data sequences. These sequences represent a variety of different shapes that can be represented using rational biquadratic spline patches.

5.1 Synthetic Sequences

The synthetic experiments were performed by generating observations of a biquadratic rational B-Spline object with control points drawn from a parabolic function. A set of 100 surface features were generated, sampled uniformly in
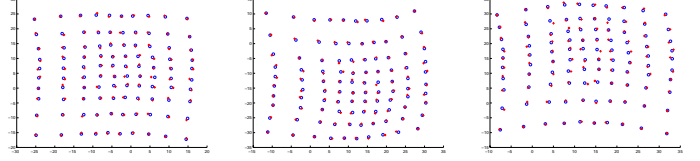
(a)Sample views. Feature observations are marked as crosses and 2D spline predicted locations marked as circles.
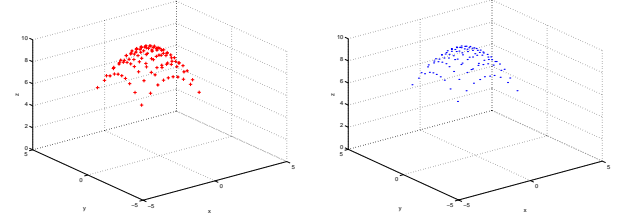
(b)Actual Object   (c) Reconstruction

Figure 1: Synthetic Reconstruction results with seven Views, 100 uniformly sampled feature points and noise with $\sigma = .4107$
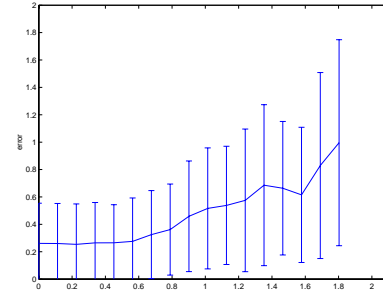

Figure 2: Average reconstruction error for $vs.$ noise level.

(s, t). The orthographic projections of the feature set were generated from seven projection matrices that corresponded to different cameras viewpoints. Gaussian noise was then added to the 2D feature locations.

Examples of the generated 2D views are shown in Fig. 1(a). From these views an estimate of the surface was extracted using our method. The estimate was then aligned with the true object by finding the rigid transform and isotropic scaling that minimized the distance between corresponding 3D points. This was necessary as the reconstruction is only defined up to a scale factor and a rigid transform. Example reconstruction results are shown in Fig. 1.

The sensitivity of the reconstruction method was tested by varying the standard deviation of the added noise. A graph of results of this experiment is shown in Fig. 2. Here we show the average reconstruction error for a given level of noise, where the error is the average distance between corresponding points of the aligned surface estimate and the true surface computed from multiple instances of the noise level. From this plot we see that our approach is reasonably stable with respect to additive noise.
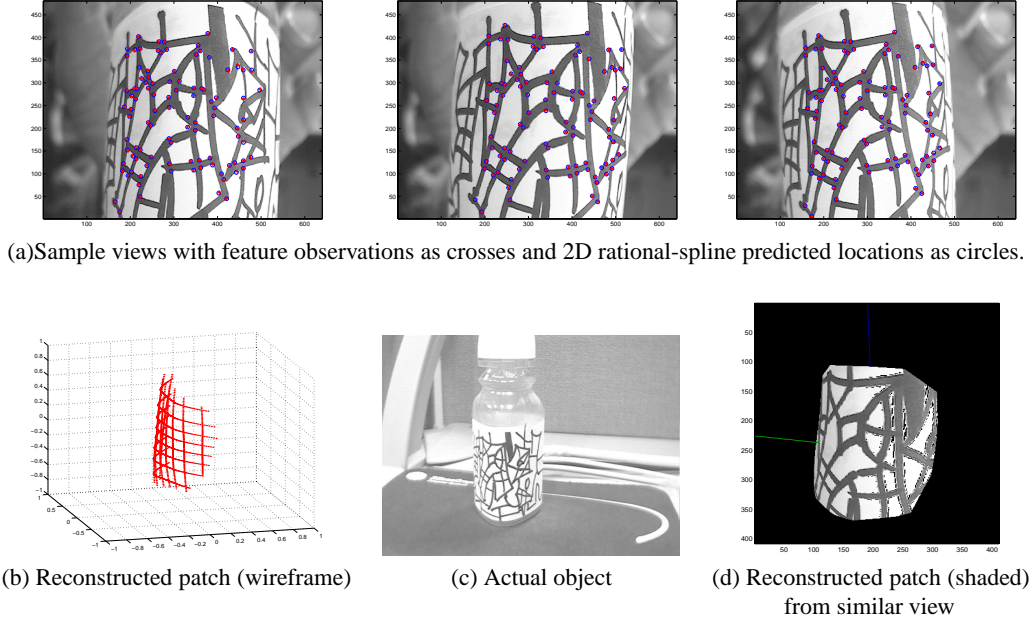
5

(a)Sample views with feature observations as crosses and 2D rational-spline predicted locations as circles.

(b) Reconstructed patch (wireframe)　　(c) Actual object　　(d) Reconstructed patch (shaded) from similar view

Figure 3: Reconstruction of bottle sequence from 101 features in four views.

5.2 Real Sequences

A number of real sequences were taken of smooth textured objects using a USB camera with resolution of 640×480. In these sequences orthographic cameras were assumed and only a patch of the object was considered for reconstruction.

Corresponding features were extracted by selecting points in one frame and tracking them with an iterative pyramidal Lucas-Kanade tracker that is available in OpenCV[1]. Following this, a few frames with disparate views were selected for use in reconstruction. Features in each selected input frame are shown in Figs. 3, 4, and 5.

In the first sequence shown in Fig. 3, four views with 101 features of the object were taken. Samples of the fitted 2D rational-Spline predicted feature locations are shown in Fig. 3(a) along with the observed feature points. Note that the cylindrical cross-section of the bottle is actually a rounded triangular shape. From this figure we see that we were able to accurately fit the 2D rational B-splines. The subsequent 3D surface reconstruction for this patch of the object, along with the an image of the actual object are shown from a similar viewpoint are shown in 3(b,c). From these plots we see the reconstructed surface appears similar to the true object, though there are some artifacts around the edges of the spline. This resulted from trimming the parts of the spline surface that were outside the convex set of observed points.

Additional example reconstructions are shown in Figs. 4 and 5. In Fig. 4 the surface of a mushroom cap was reconstructed from four views and 105 features. As seen in Fig. 4(b,c), the resulting 3D surface reconstruction accurately

includes a slight dimple in the middle of the mushroom cap surface. An example of reconstruction for a concave surface is shown in Fig. 5. In this case, seven views and 127 features were used as input to the reconstruction.

6 Discussion and Future Work

As exhibited in the experiments this approach is able to extract the shape of surfaces modeled as 3D rational spline patches. While the basic shape of the object was recovered, the visual quality of the reconstructions depends on how well features were extracted. In particular solutions exhibit oscillatory in places of the surface when there were not many or poorly tracked features. This behavior is expected, and to compensate, penalties on smoothness need to be incorporated. This can be done by adding smoothness terms in the reconstruction step [9]. Despite this, results of this method are promising.

In future work we hope to enhance this system in several ways. First, we plan to incorporate features estimation into the framework. Currently, feature estimation is a separate step. This method however, requires the use of many features and finding them is often difficult. We expect better overall performance if feature tracking is incorporated directly within surface estimation. By doing this we may better cope with occlusions and incorporate other image features, such as lines and the silhouette edges of the surface. We also plan to extend the surface representation to deal with more complex surfaces, by extending the formulation from a single spline patch to piecewise spline surfaces. Additionally, surface creases may also be considered.
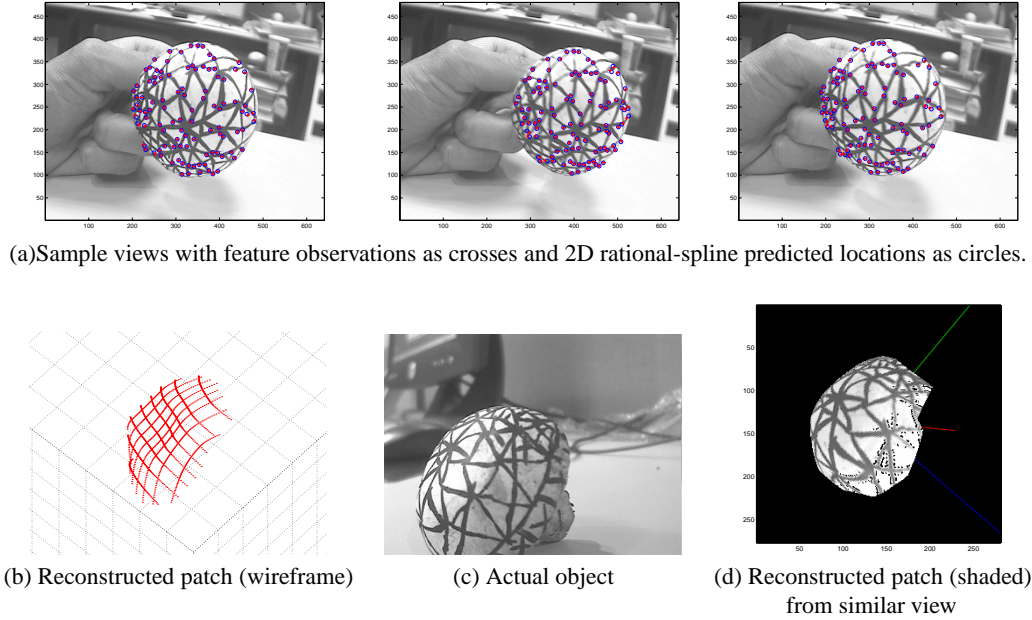
---

[1]http://www.intel.com/research/mrl/research/opencv/

(a)Sample views with feature observations as crosses and 2D rational-spline predicted locations as circles.

(b) Reconstructed patch (wireframe)

(c) Actual object

(d) Reconstructed patch (shaded) from similar view

Figure 4: Reconstruction of mushroom cap surface from 105 features in four views.

(a)Sample views with feature observations as crosses and 2D rational-spline predicted locations as circles.

(b) Reconstructed patch (wireframe)

(c) Actual object

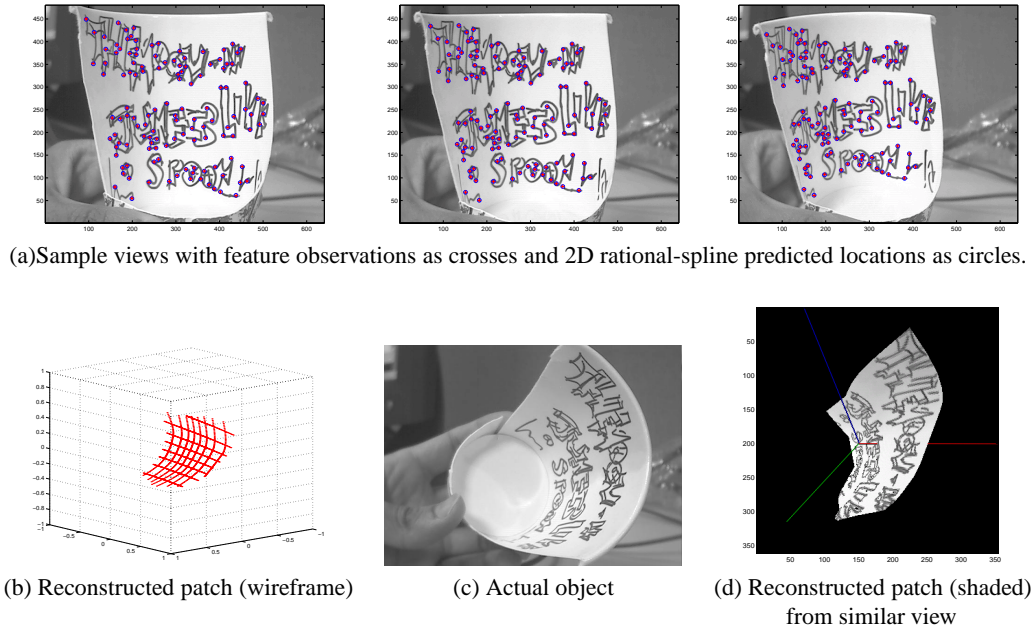(d) Reconstructed patch (shaded) from similar view

Figure 5: Reconstruction of concabe 3D rational B-spline surface from seven views and 127 features.

References

[1] J. Alon and S. Sclaroff. Recursive estimation of motion and planar structure. In *Proc. CVPR*, pages II:550–556, 2000.

[2] A. Azarbayejani and A.P. Pentland. Recursive estimation of motion, structure, and focal length. *PAMI*, 17(6):562–575, 1995.

[3] C. Baillard and A. Zisserman. Automatic reconstruction of piecewise planar models from multiple views. In *Proc. CVPR*, pages II:559–565, 1999.

[4] P.A. Beardsley, P.H.S. Torr, and A. Zisserman. 3d model acquisition from extended image sequences. In *Proc. ECCV*, pages II:683–695, 1996.

[5] A. Blake and M. Isard. *Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*, chapter 7. Springer, 1999.

[6] T.J. Broida, S. Chandrashekhar, and R. Chellappa. Recursive 3-d motion estimation from a monocular image sequence. *Trans. Aero. and Elec. Sys.*, 26(4):639–656, 1990.

[7] T. Cham and R. Cipolla. Stereo coupled active contours. In *Proc. CVPR*, pages 1094–1099, 1997.

[8] G. Cross and A. Zisserman. Quadric surface reconstruction from dual-space geometry. In *Proc. ICCV*, pages 25–31, 1998.

[9] M. Eck and H. Hoppe. Automatic reconstruction of b-splines surface of arbitrary topological type. In *SIGGRAPH*, pages 325–334, 1996.

[10] G. Farin. *Nurbs: From Projective Geometry to Practical Use*. A K Peters Ltd, 2nd edition, 1999.

[11] O.D. Faugeras and F. Lustman. Motion and structure from motion in a piecewise planar enviroment. *J. of Pattern Recognition and AI*, 2(3):485–508, 1988.

[12] G. Forsythe, M. Malcom, and C. Moler. *Computer Methods for Mathematical Computations*. Prentice-Hall, 1976.

[13] P. Fua. From multiple stereo views to multiple 3D surfaces. *IJCV*, 24(1):19–35, 1997.

[14] P. Fua and G. Leclerc. Taking advantage of image-based and geometry-based constraints to recover 3-D surfaces. *CVIU*, 64(1):111–127, 1996.

[15] J. Lombardo and C. Puech. Oriented particles: A tool for shape memory objects modeling. In *Proc. Graphics Interface*, 1995.

[16] L. Matthies, R. Szeliski, and T. Kanade. Kalman filter-based algorithms for estimating depth from image sequences. *IJCV*, 3(3):209–238, 1989.

[17] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, 1988.

[18] A. Shashua and S.Toelg. The quadric reference surface: Theory and applications. *IJCV*, 23(2):185–198, 1997.

[19] D. Sinclair and A. Blake. Quantitative planar region detection. *IJCV*, 18(1):77–91, 1996.

[20] S. Soatto and P. Perona. Recursive 3-d visual motion estimation using subspace constraints. *IJCV*, 22(3):235–259, 1997.

[21] P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *Proc. ECCV*, pages 709–720, 1996.

[22] S. Sull and N. Ahuja. Segmentation, matching and estimation of structure and motion of textured piecewise planar surfaces. In *Proc. Motion Workshop*, pages 274–279, 1991.

[23] R. Szeliski and J. Coughlan. Hierarchical spline-based image registration. *IJCV*, 22(3):199–218, 1997.

[24] R. Szeliski and H.Y. Shum. Motion estimation with quadtree splines. *PAMI*, 18(12):1199–1210, 1996.

[25] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. In *SIGGRAPH*, 1992.

[26] R. Szeliski and P. Torr. Geometrically constrained structure from motion: Points on planes. In *Proc. SMILE*, pages 171–186, 1998.

[27] L. Theiler and H. Chabbi. Facet matching from an uncalibrated pair of images. In *Proc. MVA*, 1998.

[28] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *IJCV*, 9(2):137–154, 1992.

[29] B. Triggs. Autocalibration and the absolute quadric. In *Proc. CVPR*, pages 609–614, 1997.

[30] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – A modern synthesis. In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000.

[31] J. Weng, T. Huang, and N. Ahuja. Motion and structure from point correspondences: A robust algorithm for planar case with error estimation. In *Proc. ICPR*, pages 247–251, 1988.